Tabular Presentation of the

Crypto Catalog

Version: 1.0

2017-04-19

National Information Assurance Partnership

Revision History

Version	Date	Comment
1.0	2017-04-19	Initial Catalog

Introduction

This document presents the Security Functional Requirements and Security Assurance Requirements from the *Crypto Catalog*. This tabular representation is provided for those audiences whose interest primarily lies in those portions of that document. The Protection Profile itself remains the only complete and authoritative representation, and includes discussion of assumptions, threats, and objectives.

Security Functional Requirements

ID	Requirement	Assurance Activity
FCS_COP.1.1(1)	The TSF shall perform user data encryption/decryption in accordance with a specified cryptographic algorithm [**selection**: *cryptographic algorithm*] and cryptographic key sizes [**selection**: *key sizes*] that meet the following [**selection**: *list of standards*] . The following table provides the allowed choices for completion of the selection operations of FCS_COP.1/UDE: **Application Note:** There is app note here. What we really want is a nice table of the catalog options. And also, we would like the aactivity to be eactivity.	**TSS** The evaluator shall check that the TSS includes a description of encryption function(s) used for user data encryption. The evaluator should check that this description of the selected encryption function includes the key sizes and modes of operation as specified in the table above per row. The evaluator shall check that the TSS describes the means by which the TOE satisfies constraints on algorithm parameters included in the selections made for *cryptographic algorithm* and *list of standards*. **KMDSD** The evaluator shall examine the KMDSD to ensure that the points at which user data encryption and decryption occurs are described, and that the complete data path for user data encryption is described. The evaluator checks that this description is consistent with the relevant parts of the TSS. If XTS-AES is used as the user data encryption algorithm then the evaluator shall check that the full length keys are created by methods that ensure that the two halves are different and independent. **Guidance** If multiple encryption modes are supported, the evaluator examines the guidance documentation to determine that the method of choosing a specific mode/key size by the end user is described. **Tests** The following tests are conditional based upon the selections made in the SFR. The evaluator shall perform the following test or witness respective tests executed by the developer if technically possible, otherwise an analysis of the implementation representation has to be performed. Preconditions for testing: • Specification of keys as input parameter to the function to be tested • Specification of required input parameters such as modes • Specification of user data (plaintext) • Tapping of encrypted user data (ciphertext) directly in the non-volatile memory

UDE1: AES-CBC Tests
For the AES-CBC tests described below, the plaintext, ciphertext, and IV values shall consist of 128-bit blocks. To determine correctness, the evaluator shall compare the resulting values to those obtained by submitting the same inputs to a known-good implementation.

These tests are intended to be equivalent to those described in NIST's AES Algorithm Validation Suite (AESAVS) (http://csrc.nist.gov/groups/STM/cavp/documents/aes/AESAVS.pdf). Known answer values tailored to exercise the AES-CBC implementation can be obtained using NIST's CAVS Algorithm Validation Tool or from NIST's ACPV service for automated algorithm tests (acvp.nist.gov), when available. It is not recommended that evaluators use values obtained from static sources such as the example NIST's AES Known Answer Test Values from the AESAVS document, or use values not generated expressly to exercise the AES-CBC implementation.

- **Test 1: AES-CBC Known-Answer Tests (KAT)**
  KAT-1 (GFSBox):

  To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different plaintext values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of the given plaintext using a key value of all zeros and an IV of all zeros.

  To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different ciphertext values for each selected key size and obtain the plaintext value that results from AES-CBC decryption of the given ciphertext using a key value of all zeros and an IV of all zeros.

  KAT-2 (KeySBox):

  To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the ciphertext value that results from AES-CBC encryption of an all-zeros plaintext using the given key value and an IV of all zeros.

  To test the decrypt functionality of AES-CBC, the evaluator shall supply a set of five different key values for each selected key size and obtain the plaintext that results from AES-CBC decryption of an all-zeros ciphertext using the given key and an IV of all zeros.

  KAT-3 (Variable Key):

  To test the encrypt functionality of AES-CBC, the evaluator shall supply a set of keys for each selected key size (as described below) and obtain the ciphertext value that results from AES encryption of an all-zeros plaintext using each key and an IV of all zeros.

  Key *i* in each set shall have the leftmost *i* bits set to ones and the remaining bits to zeros, for values of *i* from 1 to the key size. The keys and corresponding ciphertext are listed in AESAVS, Appendix E.

  To test the decrypt functionality of AES-CBC, the evaluator shall use the same keys as above to decrypt the ciphertext results from above. Each decryption should result in an all-zeros plaintext.

  KAT-4 (Variable Text):

  To test the encrypt functionality of AES-CBC, for each selected key size, the evaluator shall supply a set of 128-bit plaintext values (as described below) and obtain the ciphertext values that result from AES-CBC encryption of each plaintext value using a key of each size and IV consisting of all zeros.

  Plaintext value *i* shall have the leftmost *i* bits set to ones and the remaining bits set to zeros, for values of *i* from 1 to 128. The plaintext values are listed in AESAVS, Appendix D.

  To test the decrypt functionality of AES-CBC, for each selected key size, use the plaintext values from above as ciphertext input, and AES-CBC decrypt each ciphertext value using key of each size consisting of all zeros and an IV of all zeros.
- **Test 2: AES-CBC Multi-Block Message Test**
  The evaluator shall test the encrypt functionality by encrypting nine *i*-block messages for each selected key size, for $2 \leq i \leq 10$. For each test, the evaluator shall supply a key, an IV, and a plaintext message of length *i* blocks, and encrypt the message using AES-CBC. The resulting ciphertext values shall be compared to the results of encrypting the plaintext messages using a known good implementation.

  The evaluator shall test the decrypt functionality by decrypting nine i-block messages for each selected key size, for $2 \leq i \leq 10$. For each test, the evaluator shall supply a key, an IV, and a ciphertext message of length *i* blocks, and decrypt the message using AES-CBC. The resulting plaintext values shall be compared to the results of decrypting the ciphertext messages using a known good implementation.
- **Test 3: AES-CBC Monte-Carlo Test (TBD)**
  The evaluator shall test the encrypt functionality for each selected key size using 100 3-tuples of pseudo-random values for plaintext, IVs, and keys.

  The evaluator shall supply a single 3-tuple of pseudo-random values for each selected key size. This 3-tuple of plaintext, IV, and key is provided as input to the below algorithm to generate the remaining 99 3-tuples, and to run each 3-tuple through 1000 iterations of AES-CBC encryption.

```
# Input: PT, IV, Key
Key[0]=Key
IV[0]=IV
PT[0]=PT
for i = 1 to 100 {
Output Key[i],IV[i],PT[0]
for j=1 to 1000 {
if j==1 {
CT[1] = AES-CBC-Encrypt(Key[i], IV[i],PT[1])
PT[2] = IV[i]
} else {
CT[j] = AES-CBC-Encrypt(Key[i],PT[j])
PT[j+1] = CT[j-1]
}
}
Output CT[1000]
If KeySize == 128 { Key[i+1] = Key[i] xor CT[1000] }
If KeySize == 256 { Key[i+1] = Key[i] xor ((CT[999] leftshift 128) | CT[1000]) }
IV[i+1] = CT[1000]
PT[0] = CT[999]
}
```

The ciphertext computed in the 1000th iteration (CT[1000]) is the result for each of the 100 3-tuples for each selected key size. This result shall be compared to the result of running 1000 iterations with the same values using a known good implementation.

The evaluator shall test the decrypt functionality using the same test as above, exchanging CT and PT, and replacing AES-CBC-Encrypt with AES-CBC-Decrypt.

- **Test 4: UDE2: AES-CCM Tests** These tests are intended to be equivalent to those described in the NIST document, "The CCM Validation System (CCMVS)," updated 9 Jan 2012, found at http://csrc.nist.gov/groups/STM/cavp/documents/mac/CCMVS.pdf.

  Known answer values tailored to exercise the AES-CCM implementation can be obtained using NIST's CAVS Algorithm Validation Tool or from NIST's ACPV service for automated algorithm tests (acvp.nist.gov), when available. It is not recommended that evaluators use values obtained from static sources such as http://csrc.nist.gov/groups/STM/cavp/documents/mac/ccmtestvectors.zip or use values not generated expressly to exercise the AES-CCM implementation.

  The evaluator shall test the generation-encryption and decryption-verification functionality of AES-CCM for the following input parameter and tag lengths:

  Keys: All supported and selected key sizes (e.g., 128, 192, 256 bits).

  Associated Data: Two or three values for associated data length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported associated data lengths, and 2^16 (65536) bytes, if supported.

  Payload: Two values for payload length: The minimum (≥ 0 bytes) and maximum (≤ 32 bytes) supported payload lengths.

  Nonces: All supported nonce lengths (7, 8, 9, 10, 11, 12, 13).

  Tag: All supported tag lengths (4, 6, 8, 10, 12, 14, 16).

  The testing for CCM consists of five tests. To determine correctness in each of the below tests, the evaluator shall compare the ciphertext with the result of encryption of the same inputs with a known good implementation.

Security Assurance Requirements

ID	Requirement	Assurance Activity

Glossary

Common Criteria (CC)	Common Criteria for Information Technology Security Evaluation.
Extended Package (EP)	An implementation-independent set of security requirements for a category of products, which extends those in a Protection Profile.
Protection Profile (PP)	An implementation-independent set of security requirements for a category of products.
Security Target (ST)	A set of implementation-dependent security requirements for a specific product.
Target of Evaluation (TOE)	The product under evaluation.
TOE Security Functionality (TSF)	The security functionality of the product under evaluation.
TOE Summary Specification (TSS)	A description of how a TOE satisfies the SFRs in a ST.
Security Functional Requirement (SFR)	A requirement for security enforcement by the TOE.
Security Assurance Requirement (SAR)	A requirement to assure the security of the TOE.
Secure Shell (SSH)	Cryptographic network protocol for initiating text-based shell sessions on remote systems.